Testes Automatizados Usando Robot Framework – Módulo 1 – Introdução

O que é o Robot Framework?

O Robot Framework é um framework generico para automação de testes de aceitação e pode ser aplicado para implementar a técnica de ATDD. Como ele é possível realizar a automação de sistemas *web, desktop, mobile* e inclusive de *APIs*. O Robot utiliza *keywords* (palavras-chave) para definir os passos de teste.

O *framework* provê suporte para bibliotecas externas, ferramentas que são *open source* e podem ser utilizadas para a automação. A biblioteca mais popularmente utilizada com o Robot Framework é a Selenium Library que é utilizada para testes de aplicações web.

A implementação do *framework* foi feita em Python e também pode ser rodado com Jython (implementação Java do Python) e IronPython (implementação .NET do Python) além de permitir o uso nativo do próprio Python e também do Java. Algumas outras linguagens (Closure, Ruby, Perl, PHP, node.js) são suportadas via *remote interface*.

O código do *framework* é *open source* e hospedado no github.com usando a Apache License 2.0.

Foi inicialmente patrocinado pela Nokia Networks e possui uma comunidade bastante ativa.

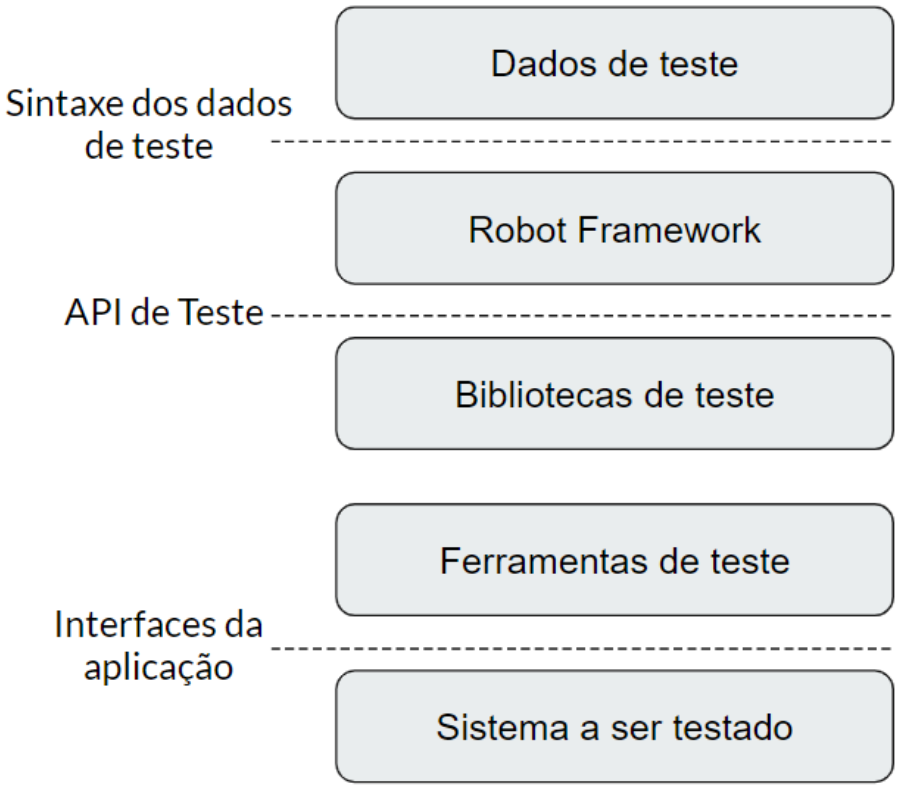
Vêm sendo utilizado largamente ao redor do mundo em diferentes domínios e contextos.

Origem e História

O *framework* foi desenvolvido na Finlândia em 2005 a partir das ideias do mestrado de Pekka Klärck. A versão 1.0 foi desenvolvida na Nokia Siemens Network e em 2008 a versão 2.0 foi liberado como *open source*. A versão no momento da escrita desse curso é 4.0 de liberada em Março de 2021.

Arquitetura

Robot Framework têm uma arquitetura modular que pode ser extendida com bibliotecas. Os dados de teste são colocados em arquivos de uma forma simples de editar e em formato tabular. Quando o Robot Framework é iniciado, ele processa os dados, executa os casos de teste e gera *logs* e relatórios da execução. O Robot não conhece nada sobre o sistema que está sendo testado, e a interação é feita através das bibliotecas.



Arquitetura do Robot Framework

Características e Recursos

Formato tabular para os casos de teste

Robot Framework vêm com uma formato tabular simples onde os casos de teste são escritos usando *keywords*. É fácil para um desenvolvedor novo entender e escrever casos de teste.

Keywords

Robot Framework provê um conjunto de palavras-chave para ser utilizado para a escrita dos casos de teste. Palavras-chave são também disponibilizadas pelas bibliotecas como a *Selenium Library* (*Open Browser*, *Close Browser*, *etc*). Também é possível criar palavras-chave definidas pelo usuário tornando a leitura do teste mais amigável para o usuário final além de poderem ser reutilizadas por outros casos de teste.

Variáveis

Robot Framework suporta o uso de variáveis dos seguintes tipos: escalares, listas e dicionário. O uso das variáveis é bastante fácil e são de grande valor ao escrever casos de testes mais complexos.

Bibliotecas

Robot Framework suporta muitas bibliotecas externas como *SeleniumLibrary*, *Database Library*, *FTP Library* e *Http Library.* A *SeleniumLibrary* é a mais usada para interagir com os navegadores web e realizar testes de sistemas web. O Robot Framework também possui suas próprias bibliotecas para manipulação de *strings*, datas, números, etc.

Casos de testes orientados pelos dados

Robot Framework suporta a definição dos casos de testes usando o estilo *keyword-driven* (palavras-chave) e o estilo *data-driven*. O estilo *data-driven* trabalha com palavras-chave de alto nível usadas como *template* para a suíte de teste e os casos de testes são usados para compartilhar dados com a palavra chave de alto nível definida no *template*. Dessa forma fica bastante fácil executar testes com diferentes parâmetros.

Marcação de casos de teste

Robot Framework permite a marcação (*tags*) dos casos de teste, assim tornando possível executar ou excluir casos de testes com determinada *tag.*

Relatórios e logs de execução

Robot Framework provê todos os detalhes da suíte de teste e da execução dos casos de teste na forma de relatórios e *logs*. Detalhes informando se o caso de teste passou ou falhou e o tempo de execução do caso de teste bem como o resultado de cada uma das *keywords* utilizadas na definição do caso de teste.

Plugins para editores

Existem diversos plugins para editores e IDEs como Eclipse, Atom, Visual Studio Code, Vim, Emacs entre outros.

Arquivo de script de teste

Um script de teste automatizado no Robot Framework com a extensão **.robot** é dividido em 4 seções:

- **Settings**: usada para configuração das bibliotecas utilizadas no teste e inclusão de arquivos auxiliares para o teste como por exemplo Page Objects
- **Variables**: lista de variáveis e seus respectivos valores utilizadas no teste
- **Test Cases**: definição dos cenários/casos de teste
- **Keywords**: definição das palavras-chave utilizadas na implementação dos cenários/casos de teste

A única seção obrigatória é a **Test Cases**, sem ela o script de teste não rodará, as demais seções são todas opcionais. Exemplo:

```
*** Settings ***
Library               SeleniumLibrary

*** Variables ***
${URL_TESTE}          https://cadastro.org

*** Test Cases ***
Fazer o cadastro de um novo cliente
    Ir para a página de cadastro de clientes
    Verificar se o título da página de cadastro de clientes está corr
    Preencher campos do formulário de cadastrado de clientes
    Cadastrar o cliente
    Verificar a mensagem de cliente cadastrado com sucesso

*** Keywords ***
Ir para a página de cadastro de clientes
    Open Browser    ${URL_CADASTRO_DE_CLIENTES}  ${NAVEGADOR_DO_TESTE
```

Selenium vs Robot Framework

Comparação de um teste automatizado escrito com o Selenium versão Java e o mesmo teste implementado com o Robot Framework

```java
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.ui.ExpectedCondition;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.WebDriverWait;

public class myclass {
  public static void main(String[] args) {
    System.setProperty("webdriver.chrome.driver",
            "C:\\selenium-java-2.35.0\\chromedriver_win32_2.2\\chromedriver.exe");
    WebDriver driver = new ChromeDriver();
    driver.get("http://www.google.com");
    WebElement element = driver.findElement(By.name("q"));
    element.sendKeys("Robot Framework Selenium Library\n");
    element.submit();
    driver.close();
  }
}
```

Versão Selenium Java

```
*** Settings ***
Library                 SeleniumLibrary

*** Test Cases ***
Abrir navegador na página do Google
    Open Browser        http://www.google.com.br   chrome
    Title Should Be     Google
    Input Text          name=q    Robot Framework Selenim Library
    Press Key           name=q    \\13
    Close Browser
```

Versão Robot Framework

Podemos ver que o código do Robot Framework é muito mais simples e fácil de ser entendido, já que todo o código executado pelo Selenium fica “escondido” na implementação das palavras-chave do Robot.

Like

Be the first to like this.